Remarks

Applicant respectfully requests reconsideration of this application. Claims 1, 6-7, 13, 16-17, and 20-21 have been amended. Claims 8-12 and 22-26 have been canceled. New claims 27-36 have been added. Therefore, claims 1-7, 13-21, and 27-36 are presented for examination.

In the Final Office Action filed July 15, 2004, claims 1-26 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Archambault (U.S. Patent No. 6,173,444) and Blainey (U.S. Patent No. 6,045,585), in view of "Restricted Pointers are Coming" by Robinson. Applicant submits that the present amendments have placed the claims in better form for allowance. Applicant further submits that the present claims are patentable over Archambault (U.S. Patent No. 6,173,444) and Blainey (U.S. Patent No. 6,045,585), in view of "Restricted Pointers are Coming" by Robinson, cited in prior Office Actions.

Archambault discloses a method that reduces the size of alias sets associated with program pointers. First, intraprocedural information about pointer variables referenced in each function of the program is gathered and saved in a data structure called a pointer alias graph. Next, the pointer alias graphs from all the compilation units for the program are combined to form a universal pointer alias graph and then transitive closure is performed on the universal pointer alias graph to produce a reduced graph containing the list of objects that each pointer variable can point to. Finally, all the files are re-compiled using the universal pointer alias graph as input, resolving all occurrences where pointer variables are de-referenced. See Archambault at Abstract.

Standard data flow gathering techniques are used to develop a pointer alias graph. The nodes in the graph represent either a definition of a pointer variable or a use of a pointer variable, and each node has an associated alias set. The initial alias sets for the nodes of the graph are defined as follows: the initial alias set for definition nodes is the right hand side of the pointer variable assignment operation, and the initial alias set for use nodes is the value of the object at that execution point (the r-val). Location information, the basic block number (relative to the flow graph) and position within the basic block, is saved for each node. In order to provide a complete representation of pointer use in the function for the interprocedural analysis, global unique names, called pseudo pointer variables, are assigned to each formal argument, function return value and global (or file scope) variable. Corresponding nodes and alias sets are created on the pointer alias graph (col. 5, ll. 4-17).

Claim 1 recites determining whether at least one pointer is aliased with at least one restricted pointer when the at least one restricted pointer is out-of-scope relative to the at least one pointer. Applicant submits that Archambault does not disclose this feature of claim 1. First, Archambault does not disclose alias analysis of restricted pointers. Second, Archambault does not disclose determining an alias with a restricted pointer that is out-of-scope relative to another pointer. The Examiner cites col. 5, lines 4-17 of Archambault as disclosing determining whether pointers are out-of-scope relative to other pointers. (See Final Office Action, page 7, part (b)). More specifically, the language cited as disclosing this feature states "a pointer alias graph is built for *each function* based on the information gathered in the first intraprocedural pass of the compiler." (Emphasis added) The word "function" as used in Archambault does not

refer to scopes within a code segment, but rather it refers to the actual procedure in a call that performs a service. Likewise, indirect function calls as referred to in Archambault describe calls that use function pointers, as the pointer indirectly references an object. (See Archambault, col. 1, lines 48-57). Therefore, claim 1 is patentable over Archambault.

The Examiner cites Blainey and Robinson as together including the feature of alias analysis of restricted pointers. Blainey discloses a system and method for determining alias information at an inter-compilation unit level of a compilation process. The method includes the steps of determining anti-alias sets from the alias information provided by a first stage of the compilation process, calculating pessimistic inter-compilation unit alias sets and refining these sets, after transitive closure as appropriate, with the anti-alias sets. See Blainey at Abstract. Nonetheless, Blainey does not disclose or suggest determining whether at least one pointer is aliased with at least one restricted pointer when the at least one restricted pointer is out-of-scope relative to the at least one pointer.

Robison discloses the uses of restricted pointers in C and C++. See Robison at paragraph 1. However, Robison does not discuss a method for optimizing C and C++ code through alias analysis that includes restricted pointers. Therefore, Robison does not disclose or suggest determining whether at least one pointer is aliased with at least one restricted pointer when the at least one restricted pointer is out-of-scope relative to the at least one pointer.

As discussed above, Archambault and Blainey do not disclose or suggest determining whether at least one pointer is aliased with at least one restricted pointer

when the at least one restricted pointer is out-of-scope relative to the at least one pointer. Furthermore, although Robinson discloses the use of restricted pointers, it does not disclose a method of alias analysis of the restricted pointers. Therefore, any combination of Archambault, Blainey, and Robinson would not disclose or suggest the features of claim 1. As such, claim 1 is patentable over Archambault and Blainey, in view of Robinson.

Claims 2-7 and 27-31 depend from claim 1 and include additional limitations. Therefore, claims 2-7 and 27-31 are also patentable over Archambault and Blainey, in view of Robinson.

Independent claims 13 and 17 contain similar features as claim 1, such as determining whether at least one pointer is aliased with at least one restricted pointer when the at least one restricted pointer is out-of-scope relative to the at least one pointer. As discussed with respect to claim 1, Archambault, Blainey, and Robinson do not disclose or suggest determining whether at least one pointer is aliased with at least one restricted pointer when the at least one restricted pointer is out-of-scope relative to the at least one pointer. As such, claims 13 and 17 are patentable over Archambault and Blainey, in view of Robinson.

Claims 14-16 depend from claim 13 and include additional limitations. Claims 18-21 and 32-36 depend from claim 17 and include additional limitations. Therefore, claims 14-16, 18-21, and 32-36 are also patentable over Archambault and Blainey, in view of Kim.

Applicant respectfully submits that the rejections have been overcome, and that the claims are in condition for allowance. Accordingly, applicant respectfully requests the rejections be withdrawn and the claims be allowed.

The Examiner is requested to call the undersigned at (303) 740-1980 if there remains any issue with allowance of the case.

Applicant respectfully petitions for an extension of time to respond to the Final Office Action pursuant to 37 C.F.R. § 1.136(a) should one be necessary. Please charge our Deposit Account No. 02-2666 to cover the necessary fee under 37 C.F.R. § 1.17(a) for such an extension.

Please charge any shortage to our Deposit Account No. 02-2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: 10/15/04

Ashley Ott

Ashley R. Ott
Reg. No. 55,515

12400 Wilshire Boulevard
7th Floor
Los Angeles, California 90025-1026
(303) 740-1980